**Task 1**

1. The class definition for Dog has been defined below.

class Dog

private name

private colour

public procedure new (myName, myColour)

name = myName

colour = myColour

endprocedure

public procedure bark(barkTimes)

for n = 1 to barktimes

print (“Woof!”)

next n

endprocedure

endclass

(a) Add new procedures to the Dog class to set the colour, get the colour and get the name of the dog.

class Dog

public function getColour()

return colour

endfunction

public procedure setColour(myColour)

colour = myColour

endprocedure

public function getName():

return name

endfunction

(b) Write statements to instantiate two new Dog objects, myDog3 and myDog4, named Mutt and Jeff, both having colour “Unknown”

myDog3 = newDog(“Mutt”, “Unknown”)

myDog4 = newDog(“Jeff”, “Unknown”)

(c) Write statements which will check the colour of one of the dogs and if it is “Unknown”, ask the user to enter its colour, and record this as the colour attribute of the dog. Print the name of the dog and its colour.

If myDog3.getColour() = “Unknown” then

myColour= input(“please enter colour for”, myDog3.getName())

myDog3.setColour(myColour)

print(myDog3.getName(), myDog3.getColour())

**Task 2**

2. Here is part of the class definition for Puppy

Class Puppy inherits Dog

private shoesChewed

shoesChewed = 0

(a) Write a method chewShoe (numShoes) which will go in the class definition of Puppy.

The method will add numShoes to shoeschewed

public procedure chewShoe(numShoes)

shoesChewed += numShoes

endprocedure

(b) Write a method getShoesChewed which will go in the class definition of Puppy.

This method will return the number of shoes chewed

public function getShoesChewed()

return shoesChewed

endfunction

(c) Write statements which will instantiate a Puppy object and call the method chewShoe several times.

Call the method getName and getShoesChewed for the Puppy object and print out the name and total number of shoes the puppy has chewed.

object = Puppy(“puppy1”, “red”)

object.chewShoe(4)

name = object.getName()

chewed = object.getShoesChewed()

print(name, chewed)

**Task 3**

3. The new class definition for Puppy is shown below.

Class Puppy inherits Dog

private shoesChewed

shoesChewed = 0

public procedure new (myName, myColour, myDob)

super.new(myName, myColour)

dob = myDob

(a) Write a new method in the Puppy class which redefines the bark method in the Dog class so that it prints “Yap!” instead of “Woof!”

public procedure bark(barktimes)

for i = 1 to barktimes

print(“woof!”)

endprocedure

(b) Instantiate a new Puppy object named Malla, colour Light brown, date of birth 12/08/2016.

Malia = Puppy(“Malia”, “Light brown”, “12/08/2016”)

(c) Call the bark method to make the puppy bark twice.

Malia.bark(2)

(d) Write a method to get the date of birth of the puppy

public function getDob()

return dob

endfunction

(e) Print out all the details of the new puppy.

print(Malia.getName(), Malia.getColour(), Malia.getShoesChewed(), Malia.getDob())

**Task 4**

4. A class called shape has subclasses named rectangle, triangle and circle.

(a) Draw an inheritance diagram showing these three classes. (Remember to use an open-headed arrow, and make sure it is pointing in the correct direction.)

**Shape**

**circle**

***Triangle rectangle***

(b) The shape class has attributes colourFill, colourOutline and a method calculateArea. Each of the inherited classes redefines the method called calculateArea, and each class has additional attributes not found in the parent class.

What is the name given to the programming language’s ability to redefine methods defined in the parent class?

inheritance

(c) The class definition for Shape is given below.

class Shape

private colourFill

private colourOutline

private area

public procedure new (myColourFill, myColourOutline)

colourFill = myColourFill

colourOutline = myColourOutline

endprocedure

public procedure calculateArea(mySide)

area = mySide \* mySide

endprocedure

endclass

class Rectangle inherits Shape

private height

private width

public procedure new (myColourFill, myColourOutline, myHeight, myWidth)

super.new(myColourFill, myColourOutline)

height = myHeight

width = myWidth

endprocedure

Complete the class definition for the Rectangle class and write the class definition for the Circle class to correctly calculate and return the area for each shape.

private colourFill

private colourOutline

public procedure new (myColourFill, myColourOutline, myHeight, mywidth)

height = myHeight

width = mywidth

colourFill = myColourFill

colourOutline = myColourOutline

class Circle inherits Shape

private radius

private colourFill

private colourOutline

public procedure new (myColourFill, myColourOutline, myRadius)

radius = myRadius

colourFill = myColourFill

colourOutline = myColourOutline

endprocedure

public function area()

return 3.14 \* (radius\*\*2)

endfunction

(d) Write statements to instantiate a rectangle and a circle and call the methods to calculate their areas. Print the areas.

rectangle = Rectangle(“red”, “blue”, 4, 5)

circle = Circle(“red”, “blue”, 3)

print(rectangle.area())

print(circle.area())